

## Contents

<b>1</b>	<b>Routine/Function Prologues</b>	<b>2</b>
1.0.1	maketiles_nongds.F90 (Source File: maketiles_nongds.F90) . . . . .	2

# 1 Routine/Function Prologues

## 1.0.1 maketiles\_nongds.F90 (Source File: maketiles\_nongds.F90)

This primary goal of this routine is to determine tile space for MPI-based I/O

### REVISION HISTORY:

```

1 Oct 1999: Jared Entin; Initial code
15 Oct 1999: Paul Houser; Major F90 and major structure revision
3 Jan 2000: Minor T=0 bug fix, should have no effect on output
8 Mar 2000: Brian Cosgrove; Initialized FGRD to 0 For Dec Alpha Runs
22 Aug 2000: Brian Cosgrove; Altered code for US/Mexico/Canada Mask
04 Feb 2001: Jon Gottschalck; Added option to read and use Koster tile space
17 Oct 2003: Sujay Kumar ; Initial version of subsetting code

```

### INTERFACE:

```
subroutine maketiles_nongds()
```

### USES:

```

use lisdrv_module, only: lis, grid, glbgindex, tile
use grid_module
use spmdMod

```

### CONTENTS:

```

if ( masterproc ) then
  if(lis%d%kgds(42) > lis%d%lnc .or. &
     lis%d%kgds(43) > lis%d%lnr) then !using a subdomain
    gnc = lis%d%kgds(42)
    gnr = lis%d%kgds(43)
  else
    gnc = lis%d%lnc
    gnr = lis%d%lnr
  endif
  lis%d%gnc = gnc
  lis%d%gnr = gnr

  allocate(lat(gnc,gnr), stat=ierr)
  call check_error(ierr,'Error allocating lat.',iam)

  allocate(lon(gnc,gnr), stat=ierr)
  call check_error(ierr,'Error allocating lon.',iam)

  allocate(mask(gnc, gnr), stat=ierr)
  call check_error(ierr,'Error allocating mask.',iam)

  nchp = lis%d%glbnch

```

```

print*, 'MSG: maketiles -- Reading ', trim(lis%p%infile), &
      ' (' , iam, ') '
open(30, file=lis%p%infile, form='unformatted', status='old')
read(30) lat
read(30) lon
read(30) mask
close(30)
print*, 'MSG: maketiles -- Done reading ', trim(lis%p%infile), &
      ' (' , iam, ') '
allocate(locallat(lis%d%lnc, lis%d%lnr))
allocate(locallon(lis%d%lnc, lis%d%lnr))
allocate(localmask(lis%d%lnc, lis%d%lnr))
localmask = 0

do r=1, gnr
  do c=1, gnc
!       if(lat(c,r).ge.36.625 .and. lat(c,r).le.36.675 .and. &
!         lon(c,r).ge.-116.025 .and. lon(c,r).lt.-115.975) then
!
!         endif
!       if(nint(lat(c,r)*1000).ge.lis%d%kgds(4).and. &
!         nint(lat(c,r)*1000).le.lis%d%kgds(7).and. &
!         nint(lon(c,r)*1000).ge.lis%d%kgds(5).and. &
!         nint(lon(c,r)*1000).le.lis%d%kgds(8)) then
!       print*, 'here ', c,r,lat(c,r),lon(c,r),mask(c,r)
!       rindex = r - (lis%d%kgds(4)-lis%d%kgds(44)) &
!         /lis%d%kgds(9)
!       cindex = c - (lis%d%kgds(5)-lis%d%kgds(45)) &
!         /lis%d%kgds(10)
!       locallat(cindex,rindex) = lat(c,r)
!       locallon(cindex,rindex) = lon(c,r)
!       localmask(cindex,rindex) = mask(c,r)
!     endif
  end do
end do
deallocate(mask)

if(lis%f%ecor .eq. 1) then
  line1 = (lis%d%kgds(4)-lis%d%kgds(44))/lis%d%kgds(9) + 1
  line2 = (lis%d%kgds(5)-lis%d%kgds(45))/lis%d%kgds(10) + 1

  allocate(elevdiff(lis%d%lnc, lis%d%lnr), stat=ierr)
  call check_error(ierr, 'Error allocating elev diff.', iam)

  elevdiff = 0.0
  print*, 'Reading elevation file ...', lis%p%elevfile
  open(21, file=lis%p%elevfile, form='unformatted', &
        access='direct', recl=4, iostat=ierr)

```

```

    if (ierr /= 0) then
        print*, "stop: problem opening elevation difference file"
        print*, "try running without elevation correction option."
        call endrun
    else
        latx = -60+lis%d%kgds(9)/(2.0*1000.0)
        lonx = -180+lis%d%kgds(10)/(2.0*1000.0)
        do r=1, lis%d%lnr
            do c=1, lis%d%lnc
                glnc = line2+c-1
                glnr = line1+r-1
                lat(c,r) = latx+(glnr-1)*0.01
                lon(c,r) = lonx + (glnc-1)*0.01
                mlat = (lat(c,r)-(latx+0.01/2))/0.01+1
                mlon = (lon(c,r)-(lonx+0.01/2))/0.01+1
                line = (mlat-1)*lis%d%kgds(42)+mlon
                read(21,rec=line) elevdiff(c,r)
            enddo
        enddo
        endif
        close(21)
        print*, 'done reading elevation difference file..'
    endif

    allocate(fgrd(lis%d%lnc, lis%d%lnr, lis%p%nt), stat=ierr)
    call check_error(ierr, 'Error allocating fgrd.', iam)

    allocate(veg(gnc, gnr, lis%p%nt), stat=ierr)
    call check_error(ierr, 'Error allocating veg.', iam)

    fgrd = 0

!-----
! Select which tile-space veg. info to use (UMD or Koster)
!-----

    print*, 'MSG: maketiles -- Reading ', trim(lis%p%vfile), &
        ' (' , iam, ')'
    open(98, file=lis%p%vfile, form='unformatted')
    read(98) lat
    read(98) lon
    do t = 1, lis%p%nt
        read(98) veg(:, :, t)
    enddo
    print*, 'MSG: maketiles -- Done reading ', trim(lis%p%vfile), &
        ' (' , iam, ')'
    do r=1, gnr
        do c=1, gnc
            isum=0.0
            if (nint(lat(c,r)*1000).ge.lis%d%kgds(4).and. &

```

```

        nint(lat(c,r)*1000).le.lis%d%kgds(7).and. &
        nint(lon(c,r)*1000).ge.lis%d%kgds(5).and. &
        nint(lon(c,r)*1000).le.lis%d%kgds(8)) then
rindex = r - (lis%d%kgds(4)-lis%d%kgds(44)) &
          /lis%d%kgds(9)
cindex = c - (lis%d%kgds(5)-lis%d%kgds(45)) &
          /lis%d%kgds(10)
do t=1,lis%p%nt
    fgrd(cindex,rindex,t) = veg(c,r,t)
enddo
end if
enddo
enddo
close(98)
deallocate(veg)

!   open (92,file='umdveg.bin',form='unformatted')
!   write(92) lat
!   write(92) lon
!   do t=1,lis%p%nt
!       write(92) fgrd(:, :, t)
!   enddo
!   close(92)

print*, 'MSG: maketiles -- Done calculations with ', &
        trim(lis%p%vfile), &
        ' (' , iam, ') '

allocate(tsum(lis%d%lnc, lis%d%lnr), stat=ierr)
call check_error(ierr, 'Error allocating tsum.', iam)
tsum = 0.0

!-----
! Exclude tiles with MINA (minimum tile grid area),
! normalize remaining tiles to 100%
!-----
do r=1,lis%d%lnr
    do c=1,lis%d%lnc
        rsum=0.0
        do t=1,lis%p%nt
            if(fgrd(c,r,t).lt.lis%d%mina)then
                fgrd(c,r,t)=0.0
            endif
            rsum=rsum+fgrd(c,r,t)
        enddo
    enddo
enddo

!-----
! renormalize veg fractions within a grid to 1
!-----
if(rsum.gt.0.0) then

```

```

do t=1,lis%p%nt
  if(rsum.gt.0.0)fgrd(c,r,t)=fgrd(c,r,t)/rsum
enddo

rsum=0.0
do t=1,lis%p%nt
  rsum=rsum+fgrd(c,r,t)
enddo

if(rsum.lt.0.9999.or.rsum.gt.1.0001)then
  write(*,*) 'Error1 in vegetation tiles',rsum,c,r
endif
endif
enddo
enddo

```

```

allocate(pveg(lis%d%lnc,lis%d%lnr,lis%p%nt), stat=ierr)
call check_error(ierr,'Error allocating pveg.',iam)

```

```

!-----
! Exclude tiles with MAXT (Maximum Tiles per grid),
!   normalize remaining tiles to 100%
! Determine the grid predominance order of the tiles
! PVEG(NT) will contain the predominance order of tiles
!-----

```

```

do r=1,lis%d%lnr
  do c=1,lis%d%lnc
    do t=1,lis%p%nt
      fvt(t)=fgrd(c,r,t)
      pveg(c,r,t)=0
    enddo
    do i=1,lis%p%nt
      max=0.0
      t=0
      do j=1,lis%p%nt
        if(fvt(j).gt.max)then
          if(fgrd(c,r,j).gt.0) then
            max=fvt(j)
            t=j
          endif
        endif
      enddo
      if(t.gt.0) then
        pveg(c,r,t)=i
        fvt(t)=-999.0
      endif
    enddo
  enddo
enddo
enddo

```

```

!-----
! Impose MAXT Cutoff
!-----
      do r=1,lis%d%lnr
        do c=1,lis%d%lnc
          rsum=0.0
          do t=1,lis%p%nt
            if(pveg(c,r,t).lt.1) then
              fgrd(c,r,t)=0.0
              pveg(c,r,t)=0
            endif
            if(pveg(c,r,t).gt.lis%d%maxt) then
              fgrd(c,r,t)=0.0
              pveg(c,r,t)=0
            endif
            rsum=rsum+fgrd(c,r,t)
          enddo
        enddo
      enddo
!-----
! renormalize veg fractions within a grid to 1
!-----
      if(rsum.gt.0.0) then
        do t=1,lis%p%nt
          if(rsum.gt.0.0)fgrd(c,r,t)= fgrd(c,r,t)/rsum
        enddo

        rsum=0.0
        do t=1,lis%p%nt
          rsum=rsum+ fgrd(c,r,t) !recalculate rsum to check
        enddo
        tsum(c,r)=rsum

        if(rsum.lt.0.9999.or.rsum.gt.1.0001)then !check renormalization
          write(*,*) 'Error2 in vegetation tiles',rsum,c,r
        endif
      endif
    enddo
  enddo
  deallocate(pveg)

  landnveg = 5
!-----
! Make Tile Space
!-----
  lis%d%glbnch=0
  do t=1,lis%p%nt
    do r=1,lis%d%lnr
      do c=1,lis%d%lnc
        if(localmask(c,r).gt.0.99.and. &

```

```

        localmask(c,r).lt.3.01)then !we have land
        if(fgrd(c,r,t).gt.0.0)then
            lis%d%glbnch=lis%d%glbnch+1
        endif
        if(tsum(c,r).eq.0.0.and.t.eq.landnveg)then
            lis%d%glbnch=lis%d%glbnch+1
        endif
    endif
enddo
enddo
enddo

print*, 'DBG: maketiles -- glbnch',lis%d%glbnch,' (',iam,')'
allocate(tile(lis%d%glbnch))

lis%d%glbngrid=0
do r=1,lis%d%lnr
    do c=1,lis%d%lnc
        if(localmask(c,r).gt.0.99 .and. &
            localmask(c,r).lt.3.01) then
            lis%d%glbngrid=lis%d%glbngrid+1
        endif
    enddo
enddo
count = 1
print*, 'DBG: maketiles1 -- glbnch',lis%d%glbnch,' (',iam,')'
allocate(grid(lis%d%glbngrid))
allocate(glbindex(lis%d%lnc, lis%d%lnr))
print*, 'DBG: maketiles2 -- glbnch',lis%d%glbnch,' (',iam,')'
do r=1,lis%d%lnr
    do c=1,lis%d%lnc
        glbindex(c,r) = -1
        if(localmask(c,r).gt.0.99 .and. &
            localmask(c,r).lt.3.01) then
            grid(count)%lat = locallat(c,r)
            grid(count)%lon = locallon(c,r)
            grid(count)%fgrd = fgrd(c,r,:)
            glbindex(c,r) = count
            count = count+1
        endif
    enddo
enddo
deallocate(locallat,stat=ierr)
call check_error(ierr,'Error allocating glblat.',iam)
deallocate(locallon, stat=ierr)
call check_error(ierr,'Error allocating glblon.',iam)
print*, 'DBG: maketiles3 -- glbnch',lis%d%glbnch,' (',iam,')'
!-----

```

```

!   For writing dominant Vegetation types
!-----
      if(lis%o%wparam .eq.1) then
          allocate(domveg(lis%d%lnc,lis%d%lnr))
          domveg = 0
      endif
      count = 0
      do r=1,gnr
          do c=1,gnc
              if(nint(lat(c,r)*1000).ge.lis%d%kgds(4).and. &
                  nint(lat(c,r)*1000).le.lis%d%kgds(7).and. &
                  nint(lon(c,r)*1000).ge.lis%d%kgds(5).and. &
                  nint(lon(c,r)*1000).le.lis%d%kgds(8)) then
                  rindex = r - (lis%d%kgds(4)-lis%d%kgds(44)) &
                      /lis%d%kgds(9)
                  cindex = c - (lis%d%kgds(5)-lis%d%kgds(45)) &
                      /lis%d%kgds(10)
                  do t=1,lis%p%nt
                      if(localmask(cindex,rindex).gt.0.99.and. &
                          localmask(cindex,rindex).lt.3.01)then
                          if(fgrd(cindex,rindex,t).gt.0.0)then
                              count = count+1
                              !
                              !
                              !
                              !
                              if((lat(c,r)-40.72).le.0.01 &
                                  .and. (lon(c,r)+77.93).le.0.01)then
                                  print*, count,lat(c,r),lon(c,r)
                              endif
                              tile(count)%row=r
                              tile(count)%col=c
                              tile(count)%index = glbgindex(cindex,rindex)
                              tile(count)%vegt=t
                              if(lis%o%wparam.eq.1) then
                                  domveg(cindex,rindex) = t*1.0
                              endif
                              tile(count)%fgrd=fgrd(cindex,rindex,t)
                              if(lis%f%ecor.eq.1) then
                                  tile(count)%elev=elevdiff(cindex,rindex)
                              endif
                          endif
                      endif
                  endif
              endif
          enddo
      enddo
!-----
! What if we we have land without vegetation assigned
!-----
      if(tsum(cindex,rindex).eq.0.0.and.t.eq.landnveg)then
          count=count+1
          tile(count)%row=r
          tile(count)%col=c
          tile(count)%index = glbgindex(cindex,rindex)
          tile(count)%vegt=t
          if(lis%o%wparam.eq.1) then

```

